Protokoll der Realisierung
Daten aus Datenbanken in der Linked Data Umgebung nutzen

Den Layer über die Datenbank legen

Mapping einer einfachen Tabelle aus Datenbank zu Linke Data

Die folgende Tabelle in einer Postgres-Datebank als RDF-Daten verfügbar gemacht werden. Es soll wo möglich das Vokabular von schema.org verwendet werden.

Vorname	Nachname	Email	Strasse	Postleitzahl	Ortschaft	Telefon	Alter
Max	Gwerder	max.gwerder@energieportal.ch	Postweg	7000	Chur	+41 44 999 21 23	54
Larissa	Odermatt	larrissa.odermatt@energieportal.ch	Im Aeuli	3007	Bern	+41 44 999 21 22	40
Luca	Studer	luca.studer@energieportal.ch	Postweg	4500	Solothurn	+41 44 999 21 25	45
Brigitte	Burkhalter	brigitte.burkhalter@energieportal.ch	Hofstrasse	6317	Oberwil b. Zug	+41 44 999 21 26	27
Veronica	Bianchi	veronica.bianchi@energieportal.ch	Bahnweg	9450	Altstätten SG	+41 44 999 21 24	27
Peter	Müller	peter.mueller@energieportal.ch	Kasernenstrasse	4052	Basel	+41 44 999 21 21	55

Installation D2RQ

Ein möglicher Ansatz dafür ist die Software D2RQ

- http://d2rq.org/

Lokale Installation

- Fork erstellt unter https://github.com/normansuesstrunk/d2rq.git
- Zip-Datei heruntergeladen und entpackt nach /home/norman/d2rq-0.8.1/
- Postgres-JDBC-Treibers bereits vorhanden unter /home/norman/d2rq-0.8.1/lib/db-drivers/postgresql-9.4-1201.jdbc4.jar

Konfiguration Postgres / Anlegen Datenbank / Benutzer / Tabelle

Dokumentation für Postgres Datenbank Ubuntu:
https://help.ubuntu.com/community/PostgreSQL

Einloggen in die Postgres-Datebank:

```
sudo -u postgres psql postgres
```

Anlegen eines Benutzer **leduser** und ihm die Datenbank **ledemployeedb** zuweisen:

- Benutzer: leduser
- Passwort: ledpassword
- Datenbank: ledemployeedb

```
sudo -u postgres createuser -D -A -P leduser
sudo -u postgres createdb -O leduser ledemployeedb
```

Datenbank löschen:

```
sudo -u postgres dropdb ledemployeedb
```

Login als leduser:

```
psql -U leduser -W -h 127.0.0.1 ledemployeedb
```

Ausführen des (SQL-Scripts)[createtable.sql] zum Anlegen der Tabelle:

```
psql -U leduser -W -h 127.0.0.1 -d ledemployeedb -a -f create-employee-
table.sql
```

Wenn man in der Datebank eingeloggt ist:

```
ledemployedb=>\i /home/norman/git/LOD_LED/d2rq/createtable.sql
```

SQL-File auf den Server kopieren:

```
scp /home/norman/git/LOD_LED/LED/mapping-db-to-ld/create-employee-table.sql
l-admin@10.0.251.93:/home/l-admin/d2rq/
```

Generieren des Mapping mit D2RQ der Datenbank zu RDF (generatemapping tool)

Das Projekt builden:

```
/home/norman/git/d2rq/# ant all
```

Datei **generate-mapping** ausführbar machen:

```
chmod +x generate-mapping
```

Mapping-Datei automatisiert erstellen, Output des Mapping in die Datei mapping-employee-postgres.ttl

```
generate-mapping -o mapping-employe-postgres.ttl -d org.postgresql.Driver -
u leduser -p ledpassword jdbc:postgresql://localhost:5432/ledemployedb
```

Daraus resultiet das Mapping-File (mapping-employe-postgres.ttl)[mapping-employe-postgres.ttl]

Starten des D2R Servers

Starten des Servers mit dem Mapping-File:

```
d2r-server mapping-employe-postgres.ttl
```

Sparql-Endpoint: http://localhost:2020/sparql

Deployen von d2rq auf Server

Issue beachten: https://github.com/d2rq/d2rq/issues/261

Datei /d2rq/src/de/fuberlin/wiwiss/d2rq/server/WebappInitListener.java anpassen, Zeile 54:

```
fileName = context.getRealPath("/WEB-INF/" + fileName);
```

Kopieren der Mapping-Datei in das d2rq web projekt:

```
cp /home/norman/git/LOD_LED/LED/mapping-db-to-ld/mapping-employe-
postgres.ttl /home/norman/git/d2rq/webapp/WEB-INF/mapping-employe-
postgres.ttl
```

Mapping-datei in /home/norman/git/d2rq/webapp/webapp/WEB-INF/web.xml eintragen. Der Pfad ist relativ zum WEB-INF Ordner:

```
....
  <context-param>
    <param-name>configFile</param-name>
    <param-value>mapping-employee-postgres.ttl</param-value>
  </context-param>
...
```

Anpassen von /d2rq/src/de/fuberlin/wiwiss/d2rq/SystemLoader.java:

```
private static final String DEFAULT_HOST = "linkeddata.fh-htwchur.ch/d2rq";
private static final int DEFAULT_PORT = 80;;
```

Dann wird auf der generierten Webseite auch der korrekte Host und die korrekten Pfade angezeigt.

war-Datei generieren

```
norman@ubuntu:~/git/d2rq$ ant war
```

war-Datei auf den Server kopieren:

```
norman@ubuntu: scp /home/norman/git/d2rq/d2rq.war l-
admin@10.0.251.93:/home/l-admin/
```

war-Datei in den Tomcat deployen:

```
root@VIRTSRVSII01: sudo cp /home/l-admin/d2rq.war /opt/apache-tomcat-
8.0.26/webapps/
```

Apache Virtual Host (/etc/apache2/sites-available/000-default.conf) anpassen:

```
#r2dq server
ProxyPass /d2rq ajp://localhost:8010/d2rq
ProxyPassReverse /d2rq ajp://localhost:8010/d2rq
```

- Applikation läuft dann unter http://linkeddata.fh-htwchur.ch/d2rq/
- Sparql Endpunkt: http://linkeddata.fh-htwchur.ch/d2rq/snorql/

Employe RDF Daten / Sparql

d2rq html views

- Alle Employes: http://linkeddata.fh-htwchur.ch/d2rq/directory/employee
- HTML View eines Employe: http://linkeddata.fh-htwchur.ch/d2rq/page/employee/1

Der Angestellte mit der ID 1 hat als Resource folgende URI: **http://linkeddata.fh-htwchur.ch/d2rq/data/employee/1**

Das Mapping ist so gemacht, dass als ein Employee auf den Employee Typ aus schema.org (http://schema.org/employee) gemapped ist. Die Spalten für Firstname/Lastname/Email/Phone sind auf die entsprechenden Eigenschaften des schema.org employee typ gemapped:

- Firstname -> givenName
- Lastname -> familyName
- Email -> email
- phone -> telephone

Die restlichen Spalten wurden in eigenem Vokabular (vocab:[spaltennamen]) gemapped.

Sparql

Liste aller Employees (im r2dq sparql endpoint müssen die prefixes nicht angegeben werden)

```
PREFIX schema: <http://schema.org/>
SELECT  * WHERE {
  ?s a schema:employee
}
```

Anzeige eines einzelnen Employe:

```
SELECT *
WHERE {
<http://linkeddata.fh-htwchur.ch/d2rq/resource/employee/1> ?p ?o
}
```

Auflisten von allen Personen:

```
PREFIX schema:  <http://schema.org/>
PREFIX vocab:   <http://linkeddata.fh-htwchur.ch/d2rq/resource/vocab/>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:     <http://www.w3.org/2001/XMLSchema#>
SELECT *
WHERE {
    ?s a schema:employee;
        vocab:employee_id           ?ID;
        schema:givenName            ?Vorname;
        schema:familyName           ?Nachname;
        schema:email                ?Email;
        vocab:employee_street       ?Strasse;
        vocab:employee_postcode     ?Postleitzahl;
        vocab:employee_city         ?Ortschaft;
        schema:telephone            ?Telefon;
        vocab:employee_age          ?Alter
}
```

Federated Query im Callimachus

http://callimachus.fh-htwchur.ch/LED/queries/AllePersonen.rq?view

Query:

```
PREFIX schema:  <http://schema.org/>
PREFIX vocab:   <http://linkeddata.fh-htwchur.ch/d2rq/resource/vocab/>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:     <http://www.w3.org/2001/XMLSchema#>
SELECT ?Vorname ?Nachname ?Email ?Strasse ?Postleitzahl ?Ortschaft ?Telefon
?Alter
WHERE {
    SERVICE <http://linkeddata.fh-htwchur.ch/d2rq/sparql> {
        ?employee_resource a schema:employee;
        vocab:employee_id       ?ID;
        schema:givenName        ?Vorname;
        schema:familyName       ?Nachname;
        schema:email            ?Email;
        vocab:employee_street   ?Strasse;
        vocab:employee_postcode ?Postleitzahl;
        vocab:employee_city     ?Ortschaft;
        schema:telephone        ?Telefon;
        vocab:employee_age      ?Alter
    }
}
```